Mississippi State UNIVERSITY

①

Center for Air Sea Technology

DESIGN OF AN INTELLIGENT SUPPORT SYSTEM FOR SCIENTIFIC DATABASES

by

Patrick Perrin and Frederick Petry

Technical Report 94-2

1 August 1994

Prepared for: Office of Naval Research (Code 1242)
Under Research Grant N00014-92-J-4109

TECHNICAL REPORT 94-02

DESIGN OF AN INTELLIGENT SUPPORT SYSTEM FOR SCIENTIFIC DATABASES

by

Patrick Perrin[1] and Frederick Petry[2]

[1]Graduate Research Assistant, Mississippi State University Center for Air Sea Technology and Graduate Research Assistant, Tulane University Department of Computer Sciences, New Orleans, LA 70118

[2]Research Affiliate Mississippi State University Center for Air Sea Technology and Professor, Tulane University Department of Computer Science, New Orleans, LA 70118

1 August 1994

This research was supported by the Department of the Navy, Office of the Chief Naval Research under Research Grant N00014-92-4109 with Mississippi State University. The opinion, findings, conclusions, and recommendations expressed in this publication are those of the authors and do no necessarily reflect the views of the U.S. Government. No official endorsement should be inferred.

ABSTRACT

The present technical report describes current progress made in designing an Intelligent Support System (ISS) for automatic data analysis (i.e., knowledge acquisition) and efficient data exploitation in numerical scientific databases, with an application to the CAST-NEONS environmental databases used for ocean modeling and prediction. First, we explain the improvements made to the Automatic Data Analysis System (ADAS) by combining the features of two machine learning techniques (i.e., data clustering and inductive learning by decision tree) to generate sets of production rules that efficiently describe the observational raw data contained in the scientific databases. Data clustering allows the system to classify the raw data into clusters, which the system learns by induction to build the decision trees, from which are deduced the production rules. Second, we describe the design of a robust computational model of conversation (ENTRETIEN) for interactive natural language man-machine interfacing, based on case grammar theory. Such a system allows the user to naturally communicate with the system for efficient and effective information retrieval. ENTRETIEN conducts the conversation to recognize the user's intentions (i.e., user's plans). To include the automatically generated production rules into the knowledge base of an expert system and to add the facilities of the natural language interface represent our final goal in building an intelligent support system for efficient exploitation of large scientific databases.

1 Needs for an Intelligent Support System

Scientific databases constantly and rapidly grow in terms of the amount of gathered knowledge. This huge flood of raw knowledge is mainly numerical data representing observations (e.g., satellite infrared images, ship and storm tracks, etc.). It is predicted to be of the order of gigabit or terabit of new raw data per day. These fast growing scientific databases render impossible, even unfeasible, human raw data analysis; hence the urgent need for automatic data analysis [PSF91] and efficient human-machine interface.

Our research project responds then to the need to find knowledge in the flood of data and to efficiently allow the user to access and use this knowledge. It is then concerned in filling the gap between data generation and data understanding for efficient and effective data exploitation, such as retrieval of information and inferences on current knowledge. Our approach is for the system to discover, in an automatic manner, relationships among the objects contained, or to be inserted, in the databases. This corresponds to finding interesting and useful patterns in the raw observational data stored in the databases. Such patterns represent an effective and efficient description of the data. Transforming those patterns into production rules to be inserted in the knowledge base of an expert system, and adding the facilities and power of an interactive conversational natural language interface, will allow the system to do efficient exploitation of large scientific databases. Designing and implementing such an Intelligent Support System (ISS) is our final goal in this research. The overall ISS functional components are depicted in Figure 1.

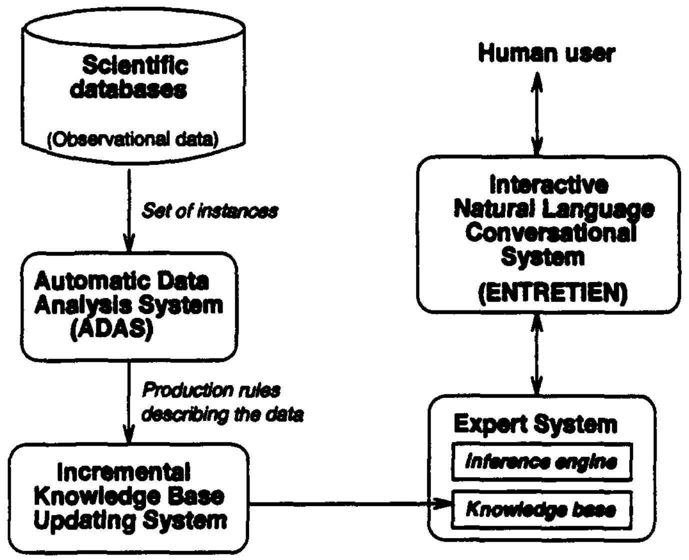
Scientific databases
(Observational data)
Set of instances
Automatic Data Analysis System (ADAS)
Production rules describing the data
Incremental Knowledge Base Updating System
Human user
Interactive Natural Language Conversational System
(ENTRETIEN)
Expert System
Inference engine
Knowledge base

Figure 1: ISS functional components

The second section of this report is concerned with the discovery-learning process of ISS. It describes ADAS, an Automatic Data Analysis System, which combines the strengths of two machine learning techniques, that are data clustering and induction learning by decision trees. The intent is to generate production rules that describe the raw data contained or being included in the scientific databases. It continues and enhances previous work in which a system for knowledge discovery in scientific databases was designed and implemented using data clustering, classification heuristics, and domain knowledge [PP94]. This previous system analyzed the raw data in order to find structure in the data. This was done by partitioning the objects space into a hierarchy of clusters, each representing a class of objects that are considered to describe a similar concept. The clustering was performed using some pre-defined classification criteria (such as object features), some heuristics, and some domain knowledge. The present research adds to this basic system a set of algorithms to make a general description of each cluster. Such descriptions are expressed in first order predicate logic as production rules. These production rules are intended to be used by an expert system for efficient exploitation of the knowledge contained in scientific databases (e.g., inference and information retrieval). These algorithms are based on inductive learning by decision trees and automatic rule generation and were implemented by Quinlan in the C4.5 package [Qui93]. The remainder of the second section presents the results of experiments conducted in applying ADAS to CAST-NEONS, a real world scientific database currently used by the Center for Air Sea Technology (CAST) for ocean modeling and prediction. By these experiments, it is shown that the system was able to identify automatically, from scratch, the Gulf Stream in the North Atlantic area by analyzing and classifying measures of its features, such as salinity and temperature. This was done at constant depth but we intend to enable the system to search more interesting correlations such as functions of salinity, temperature, depth, and temporal variables. By these examples, we empirically demonstrate the feasibility of our approach in doing automatic numerical data analysis.

The third section discusses the effective exploitation of the knowledge by the user. Discoveries are appropriately formatted by the system for the intended user. In the first stage, this is done as first order predicate logic rules obtained from decision trees. This is a convenient knowledge representation for knowledge manipulation and analysis, and is easily transformable into natural language for human user data exploitation. This section discusses the design of an interactive and robust natural language conversational simulation model we named ENTRETIEN. Since language is a natural support for human being communication for transfer of information, ENTRETIEN will allow the human user to naturally communicate with the system to efficiently exploit the potential information non-trivially stored in the large scientific databases. This is to palliate the actual obvious lack of efficient and effective communication between human users seeking information and computer systems willing to provide such knowledge. ENTRETIEN will conduct the dialogue with the user by analyzing each user's utterance (e.g., query, command, etc.) expressed in a natural way (e.g., English) and in the current context of the conversation to better serve the user's needs. This is based on human natural language information seeking dialogues during which each participant attempts to recognize the other participant intentions or plans for successfully communicating information. Our system will thus interactively conduct the dialogue to recognize the user's intentions in seeking information in the scientific numerical databases. We want to base our plan recognition simulation model on a previous model developed by Carberry [Car90]. This model seems to be the most appropriate current computational simulation model for further enhancements towards a more robust system that will be a step closer to the human way of handling information seeking dialogues. Such model of conversation is based on standard natural language processing models that simulates human cognitive processes in understanding and generating utterances in natural language such as English. Our natural language system (NLS) is based on case grammar [Fil68] with case frames in a semantic conceptual network paradigm. Such NLS is enhanced with other cognitive processes as well as pragmatics and planning strategies to efficiently determine the meaning of each current utterance in the context of the actual dialogue and to recognize the plans of the user dialogue participant.

In the last sections, this report presents future directions toward the goal of designing and implementing the intelligent support system. This includes the design of the knowledge base updating system, the integration of the productions rules and a previous semantic network that captures the domain knowledge into an expert system knowledge base, the implementation of ENTRETIEN for the system to naturally converse with human users, the selection of better data clustering algorithms to refine and render more robust the knowledge discovery/learning process, and finally to encode the knowledge found into a generic form suitable for the knowledge base.

2 Discovery-Learning Process

This section describes the discovery and learning processes of the Intelligent Support System (ISS) implemented in the Automatic Data Analysis System (ADAS). Scientific databases mainly contain raw data corresponding to observations of some parameters. Knowledge discovery in such databases is done by analyzing and classifying observations into clusters to obtain "implicit, previously unknown, and potentially useful" new information [PSF91]. New information, or patterns, are interesting and certain enough descriptions of the relationships among clusters of observational data as well as descriptions of the main characteristics of each cluster. To determine whether a pattern is interesting is based on criteria imposed by domain and background knowledge. In addition, it has to be novel, useful regarding the domain of application, and the discovery process must be non-trivial (i.e., the system takes autonomous decision on how to process the data and evaluate the results). Piatetsky-Shapiro and Frawley added that knowledge is considered useful if it facilitates the system, or the user, to satisfy a given goal [PSF91]; For example, identifying the Gulf Stream features in the North Atlantic to infer predictions (see application examples in Section 2.5). Patterns are described using first order predicate logic (i.e., production rules) because it is well known that it is a deterministic manner to encode knowledge. It can also be easily transformed into a high level language, such as a subset of English, to be understood by human users, which is the main goal in discovering knowledge. Extraction of knowledge by ADAS from non-trivially stored raw observational data is mainly done in two steps:

- First, the system must identify interesting patterns underlined in the raw data. This is called the **identification process** [PSF91], during which ADAS clusters the instance space into subclasses that reflect regularities inherent in the observational data. Our approach is to use multivariate data analysis and unsupervised machine learning techniques, such as data clustering. This phase follows a prior study [PP94] in which we used mainly statistical data analysis (e.g., data clustering techniques) to discover hidden knowledge, or useful regularities in the data (see also [ZB91] and [PSF91]). The previous classification used the Euclidean distance in a N-space among objects' features to cluster similar objects into a hierarchy of classes. The search in what to look for is guided by heuristics, domain knowledge, and the user's current interests (e.g., Gulf Stream features, as shown in the experiments).
- Second, the system must describe these clusters into a concise and meaningful representation for the intended user, such as first order predicate logic production rules for use by an expert system shell or natural language sentences for use by a human user. This phase is called the **descriptive process** [PSF91], and it summarizes the relevant features that describe each cluster. This is related to supervised machine learning techniques.

One of the main issues of developing ADAS is to use computationally efficient algorithms to deal with large volumes of data and to use some domain knowledge and heuristics to guide the search for hidden information.

The Automatic Data Analysis System is composed of three main components as depicted by Figure 2. The dataset of training instances (a selection of raw observational data), which forms the N-dimensional data space, is first statistically analyzed or searched for structure (identification process). This is done by data clustering. The data space is partitioned into clusters (or classes) such that there is both maximal dissimilarity between clusters and maximal similarity between instances in each cluster. Since these two characteristics are contradictory, trade-offs must be found. These clusters become classes of the instances for the supervised learning phase (inductive learning). In the inductive learning process, each cluster is described by a decision tree (descriptive process), from which production rules are generated. Each of these components is now described.

Following the requirements to design systems for knowledge discovery in databases that were suggested during the workshop on Knowledge Discovery in Databases at the Eleventh International Joint Conference on Artificial Intelligence [PSF91], the ADAS includes the four following characteristics:

1. **High-level language**. To be considered knowledge, findings must be understood by both machine and human, hence the use of production rules that can be transformed into well understood English sentences (see the design of ENTRETIEN in Section 3).

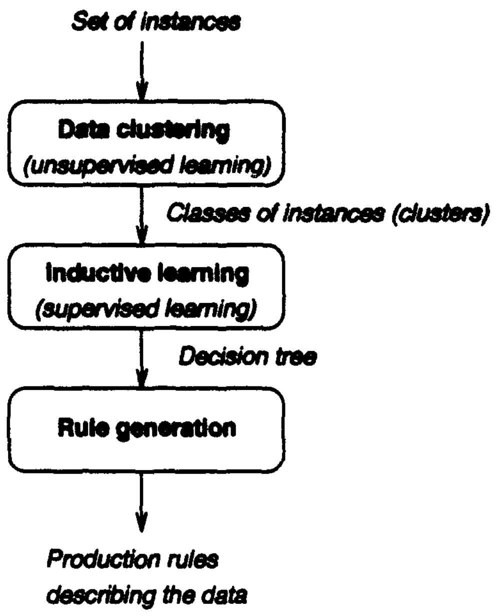

Figure 2: ADAS' functional components

2. **Accuracy**. Certainty factors are attached to data to ensure the reliability in the findings.
3. **Interesting patterns**. Findings must be evaluated "interesting" to be kept in the knowledge base as new knowledge of interest for the application domain. This is decided according to a combination of heuristics and domain knowledge.
4. **Efficiency**. The discovery process should be predictable and computationally acceptable when analyzing large amount of data.

A discussion of those characteristics is considered throughout the following subsections, but we first describe the functional components that compose ADAS as well as the strategies used to analyze the data. Then, we discuss a simple approach to handle uncertainty in the discoveries, and finally, we present experiments conducted on gridded data from the CAST-NEONS scientific databases.

2.1 Data clustering: finding structure in the data.

This section briefly summarizes previous work done by Perrin and Petry [PP94]. Multivariate data analysis, also called data clustering, is intended to analyze the statistical behavior of the objects of a given set to get complete information on them. It facilitates greatly human comprehension of raw observational data. It is believed that with enormous volumes of data, one can rely on statistical analysis methods to describe them [Gre85].

2.1.1 General approach.

Our approach to describe the set of raw observational data uses data clustering to find structure (functional relationships) in the data. Such structure is used to hierarchically classify the instances (see Figure 3). From this hierarchy are derived a number of clusters, each containing objects that describe the same concept, according to some criteria, such as object features. Some domain knowledge and heuristics are used during the data clustering process to avoid finding unnecessary relationships between every pair of objects and to keep high computational efficiency. This is justified by the fact that some relations are known not to be of interest for the particular application, and thus the system saves computational effort and time by not processing them. This is also to palliate the problem of deciding which features are to be taken in account for clustering [PP94]. Similarity between instances is done by computing the Euclidean distance between every pair of attributes. Note that this resemblance coefficient (i.e., Euclidean distance) handles missing values in the data matrix. The larger the degree of similarity (i.e., distance) is between two instances, the less similar these instances are. The obtained resemblance data matrix, which groups similarity measures between every pair of instances, is applied to a clustering algorithm that builds a hierarchical tree of the instances. This tree is then analyzed and interpreted to decide where to cut it; or to decide how many clusters have to be considered (see Section 4.1).

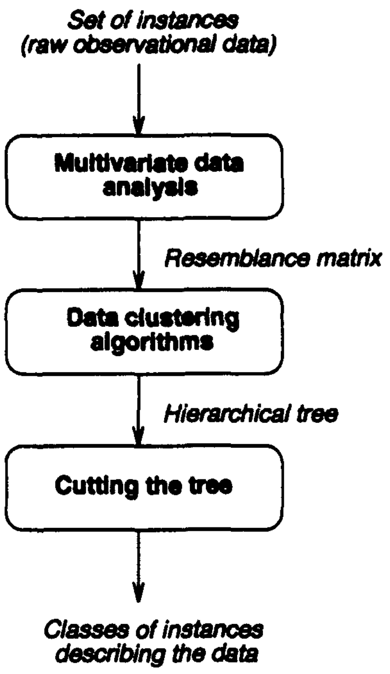
Set of instances
(raw observational data)
Multivariate data analysis
Resemblance matrix
Data clustering algorithms
Hierarchical tree
Cutting the tree
Classes of instances describing the data

Figure 3: Data clustering process

2.1.2 Implementation.

Multivariate data analysis and data clustering. We used the Cluster 2.7 package designed by Yoshiro Miyata and Andreas Stolcke at Berkeley University, and freely distributed [MS93]. The clustering algorithm used in this package is a variant of the Single Linkage Clustering Analysis (SLCA). According to Stolcke, the main difference between Cluster 2.7 and SLCA is that inter-cluster distances are computed using the cluster-means rather than the minimal distance between cluster members (see Section 4.2).

Determining the number of clusters. To use C4.5 on clustered objects, we needed to come up with a way of assigning classes to the clusters produced by Cluster 2.7. The approach taken was to restrict the number of classes to a power of 2. The hierarchy produced by Cluster 2.7 is a binary tree with its leaves representing the objects extracted from the scientific database. Given the binary tree structure produced by Cluster 2.7, the class assignment algorithm can easily traverse the tree in an in-order method and easily add objects to a class by simply determining its current level. If the current level is greater than $\log_2(number\_of\_classes)$ then it would add the object to the current class. If the algorithm moves to a level which is less than $\log_2(number\_of\_classes)$ and then traverse back down then it would increment the class by 1 and start adding objects to the new class. The algorithm continues in a recursive manner until it has traversed the entire tree.

Although this approach is very fast and straightforward, it has a tendency to produce classes with minimal amounts of information. In the examples mentioned later, it will be seen that class C1 covers only a small region of the grid compared to class C2 or C3. A better algorithm would put C0 and Cl in the same class and keep C2 and C3 as separate classes. We intend to explore a different approach to perform this task in future work (see Section 4.1).

2.2 Decision trees - describing the clusters.

2.2.1 General approach.

Although the Cluster 2.7 package is very powerful at finding clusters, one can not produce a set of rules from or extract information about the clusters. The C4.5 machine learning package, developed by Quinlan [Qui93], will produce decision trees that production rules can readily be extracted from. Like Cluster 2.7, C4.5 performs classification. A two step process using Cluster 2.7 and C4.5 is needed because C4.5 requires pre-defined and discrete classes to perform supervised learning. Cluster 2.7 provides the clustering information needed to derive pre-defined and discrete classes.

C4.5 builds decision trees in a divide and conquer approach based on E.B. Hunt's method for constructing decision tree from a set of training cases [Qui93]. The approach is simple and works as follow. Using a set T of training cases, let the classes be denoted $\{C_1, C_2, C_3, ..., C_k\}$. There are three possibilities for the training cases:

1. All of the training cases belong to a single class C_j. Thus, the decision tree is a single leaf identifying class C_j.
2. The training set T contains no cases. Hence, the decision tree is just a leaf but the class associated with the leaf is determined from information outside of the training cases.
3. T contains cases belonging to many classes. In this case, the goal is to head towards single-class collections of cases. A test, based on a single attribute, is chosen so that there are mutually exclusive outcomes $\{O_1, O_2, O_3, ...O_n\}$. T is then partitioned into subsets $T_1, T_2, T_3, ..., T_n$, where T_i contains all the cases of T that have the outcome O_i of the test. This is done recursively on each subset T_i [Qui93].

2.2.2 Tests to partition test cases.

Finding tests to partition the test cases is an important task. The purpose is not to simply partition the test cases but to help build a decision tree that accurately reflects the domain structure and the predictive powers associated with knowing the properties of the domain. Ideally, tests should be chosen at each stage to produce a small final tree [Qui93].

Unfortunately, as Hyafil and Rivest showed, finding the smallest decision tree consistent with a training set is NP-complete [Qui93]. The method used in C4.5 to construct decision trees is a non backtracking, greedy algorithm. When a test has been selected to partition the current set of cases, the choice is set and alternative choices are not explored.

C4.5 uses a gain ratio criterion for evaluating tests. The gain ratio criterion is a modification of the gain criterion used in the ID3 algorithm (cf. Appendix 7.1). Quinlan found that the gain criterion had a strong bias which favored tests with many outcomes. Quinlan uses the hypothetical situation of a medical diagnosis task where one of the attributes happens to contain a unique patient identification. Partitioning the training cases based on this attribute will produce a large number of subsets containing one case. The gain ratio criterion approach fixes this bias by normalizing the gain attributed to tests with many outcomes. This is done using the equation $split\_info(X) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \times \log_2\left(\frac{|T_i|}{|T|}\right)$ which represents the potential information generated by dividing T into n subsets, while the information gain measures the information relevant to classification that arises from the same division [Qui93]. Now, $gain\_ratio(X) = \frac{gain(X)}{split_info(X)}$ expresses the proportion of information that is helpful for classification. To avoid unstable ratios produced by trivial splits, the gain ratio criterion chooses a test that will maximize this ratio with the constraint that the information gain is large [Qui93].

In the medical diagnosis example previously mentioned, Quinlan states that the patient identification attribute will not be ranked highly using the gain ratio criterion.

2.3 Generation of production rules.

The process of generating production rules from a decision tree is a straightforward process. Each path in the decision tree corresponds to an initial rule. The left hand side of the rule is the set of conditions that follow the path along the tree to the leaf. The right hand side of the rule represents the leaf. Conditions are removed from each rule that do not help distinguish the selected class from the other classes using a pessimistic estimate of rule accuracy. The next step is to delete simplified rules that do not contribute to the accuracy of the set of rules. After this step, the simplified rules are ordered to minimize false positive errors. A default class is chosen also during this last step [Qui93].

2.4 Uncertainty in the findings.

Certainty is a measure of the degree of belief the system has in the findings. This is an important issue in knowledge discovery that must be considered to assess the credibility and the quality of the findings, as Piatetsky-Shapiro and Frawley noted "without sufficient certainty, patterns become unjustified and, thus, fail to be knowledge" [PSF91]. Each object and discovery in ADAS is associated with a computed certainty factor (CF), corresponding to the certainty in the data. It measures the integrity of the data objects used for the discovery. Data in a database can be missing, noisy, or inapplicable, and this influences on the reliability of the findings. The certainty factor approach is justified by the fact that it is well suited when the knowledge is in rule format, and it is a simple computational model efficient enough for the CAST scientific database [GD93].

2.4.1 General approach.

From the database data dictionary, a specific feature of an object is expected to have a certain content (e.g., value, etc.). When such a content is controversial, this has to be reflected on the certainty of the whole object. First, our approach is to consider the object to be fully reliable (i.e., certainty CF=1.0). Second, this certainty is modified when analyzing each characteristic of the object to determine its validity (e.g., missing fields, improper value, etc.).

Data in CAST-NEONS database are mainly observations. There is no background knowledge (i.e., domain knowledge) that can be used to compute the confidence factor in the object. Hence, we use the following simple approach. The object data certainty is computed from each of its N feature certainties (CF_i). Features are composed of main features and sub-features, with the sub-feature certainties contributing to their parent feature. For example, a grid object has 8 main features, that are model, geometry, version, base etm, parameter, level type, level value, and grid values. The main feature grid values are a series of grid measures (each being a sub-feature). The contribution (C_i) of each of the N features is $C_i = \frac{CF_i}{N}$. The data certainty is $CF = \sum_{i=1}^{N} C_i$, where $-1 \leq CF \leq 1$. Note that CF=-1 means a maximum disbelief on the data, CF=0 means that it is not possible to decide whether data is reliable or not, and CF=1 means a maximum belief on the data. CF_i is the certainty of feature N_i and is determined as follows:

$CF_i = 1.0$ (maximum belief) if the feature content is what is expected (valid type, range, etc.). This is determined by checking the background knowledge (e.g., database data dictionary) that contains the description and constraints on each attribute.

$CF_i = 0.0$ (cannot decide) if content is null (case of missing value). There is no evidence to decide whether there is belief or disbelief of this feature.

$CF_i = -1.0$ (maximum disbelief) if content is not what is expected (case of inappropriate or noisy data).

Note that when feature N_i is divided into M sub-features, then $CF_i = \frac{1}{M} * \sum_{j=1}^{M} CF_{i,j}$.

2.4.2 Example.

The following example illustrates how to compute the certainty CF in the data. Let a grid object be defined by N=8 main features, feature 8 being defined by M=15000 sub-features (each is a measure at a given longitude and latitude of the grid). The following table defines each feature:

i	Feature	Content		CF_i	C_i
1	model	DYN-CLMO	valid content	1.0	1/8
2	geometry	xyz48	not valid (unknown to background knowledge)	-1.0	-1/8
3	version	null	missing value	0.0	0.0
4	base etm	170088	valid content	1.0	1/8
5	parameter	salinity	valid content	1.0	1/8
6	level type	sgma-lvl	valid content	1.0	1/8
7	level value	6.	valid content	1.0	1/8
8	grid values	series of 15000 values	suppose there are 2 values out of range (noisy data)	14998/15000 = 0.9998666	0.124

Hence, the overall certainty on the data of this grid object is CF=0.624, and it represents the degree of reliability the system has on this grid object.

2.5 Application to CAST/NEONS databases - Examples.

2.5.1 CAST-NEONS databases and grid data.

The Center for Air Sea Technology (CAST) manages large scientific environmental databases called CAST-NEONS. The implementation is based on the Empress Relational Database Management System (Empress RDBMS) [3] interfaced with the Naval Environmental Operational Nowcast System (NEONS) [4]. NEONS eases the management of the Empress RDBMS, allows one to query the databases via embedded-SQL C functions, and facilitates the display and the analysis of environmental data.

CAST-NEONS databases also contain environmental data from National Oceanic and Atmospheric Administration (NOAA) and Navy archives. It is used for research and evaluation of ocean modeling and prediction, as well as ocean features studies such as eddy resolution, currents, surface structure, etc. [NOA92]. The environmental data is composed of five generic data types:

- **Image data**: satellite multi-band images and image overlays.
- **Latitude-Longitude-Time data (LLT)**: specific measures (from buoys, ships, etc.) of one or more parameters established by position (latitude and longitude) and time.
- **Line data**: a series of observational points along a (curved) line representing storm tracks, etc.
- **Grid data**: values of measures at coordinates established by a grid of points at a given time and level. Grid data are 4-dimensional: latitude, longitude, time, and parameter measured.
- **Volume data**: 5-dimensional set of related grid data.

The present study will focus on the grid data type. This choice is a good trade-off in using a sufficiently complex data type to show the feasibility of the study and to develop a quick prototype. Grid data contain atmospheric and oceanographic numerical outputs, user-defined products, and grided climatology. They are organized into model types. Each grid is stored in the databases by grid field, i.e., one parameter at a given time for all the points of the grid. A grid instance is uniquely identified by the following eight fields:

- Grid model type
- Grid model version name
- Grid model geometry name

[3] ©Empress Software, Inc.
[4] Developed by NOARL

- Geophysical parameter name measured
- Level type at which the parameter is measured
- Level value at which the parameter is measured
- Base time (date and hour) when the measured were taken
- Forecast period

In addition to these fields, there is associated a bit stream of values to the grid. It is variable in length for each grid and depends on the grid model. It is the set of actual measure values of the given parameter at the given level for every position (latitude and longitude) determined by the grid model.

2.5.2 Example from CAST/NEONS: salinity and temperature in the Gulf Stream - General approach.

We present here three of the experiments we conducted to test ADAS. They correspond to gridded data in the Gulf Stream (North Atlantic area) from three consecutive days (day 1: July 15, day 2: July 16, and day 3: July 17 of 1991) at a constant depth (2.5 meters below the sea surface). The goal is for the system to automatically determine Gulf Stream features only by studying the correlation of depth, salinity, and temperature, at given grid locations (longitude, latitude).

To do so, some training instances corresponding to the grid model, dates and depth and measurements of salinity and temperature were selected from CAST-NEONS databases (see Figure 4 for the training instance object template definition). Both salinity and temperature grids were combined, then divided into 15000 small cells that represent a training instance.

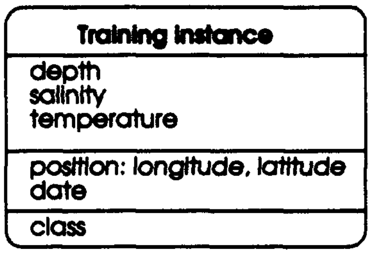

Figure 4: Training instance template

Unsupervised learning (i.e., data clustering) allows one to determine to which class a training instance belongs to. Applying the Cluster2.7 algorithms on the instance space produces a hierarchy of instances classified by how close they are from each other in the space (we used Euclidean distance). Deciding arbitrarily on the number of clusters to keep (four for the present three cases), we were able to assign a class to each training instance. Applying the C4.5 algorithms on the training instance dataset resulted in decision trees used to generate production rules. Each of the three experiments shows:

- A 2D representation of the instance space (i.e., grid) that indicates which instance were grouped together by data clustering. In each case, the system classified the data into four main clusters (cf. Figures 5, 8, and 11).
- Two 3D representations for the salinity (cf. Figures 6, 9, and 12) and the temperature (cf. Figures 7, 10, and 13) showing graphically the role of each cluster in terms of the parameter measured.
- The decision tree obtained with C4.5 that describes logically the information contained in the previous pictures (cf. appendices 7.2, 7.4, and 7.6).
- And finally the production rules that were generated from the decision tree. These rules describe the findings of the system (cf. appendices 7.3, 7.5, and 7.7).

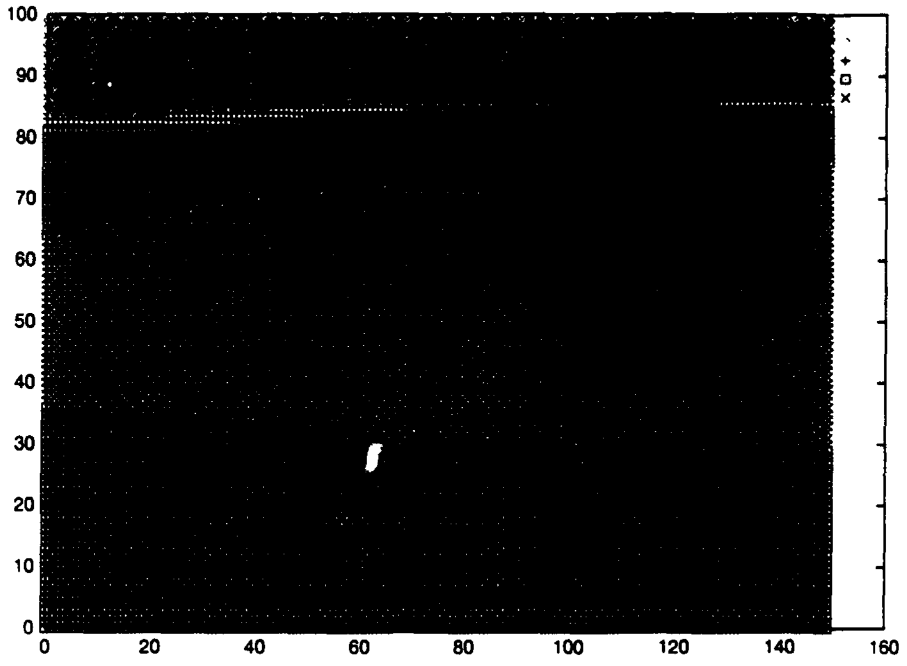

Figure 5: Day 1: Instance space 2D representation

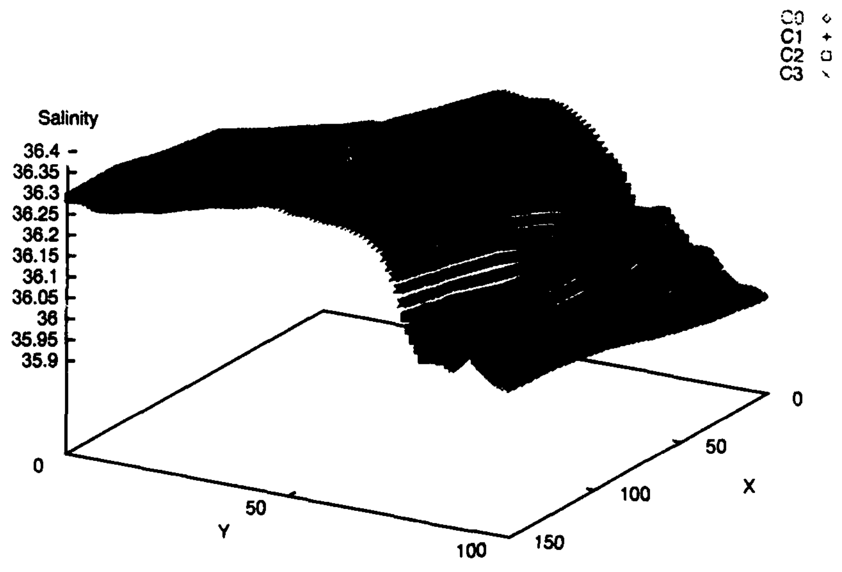

Figure 6: Day 1: Salinity 3D representation

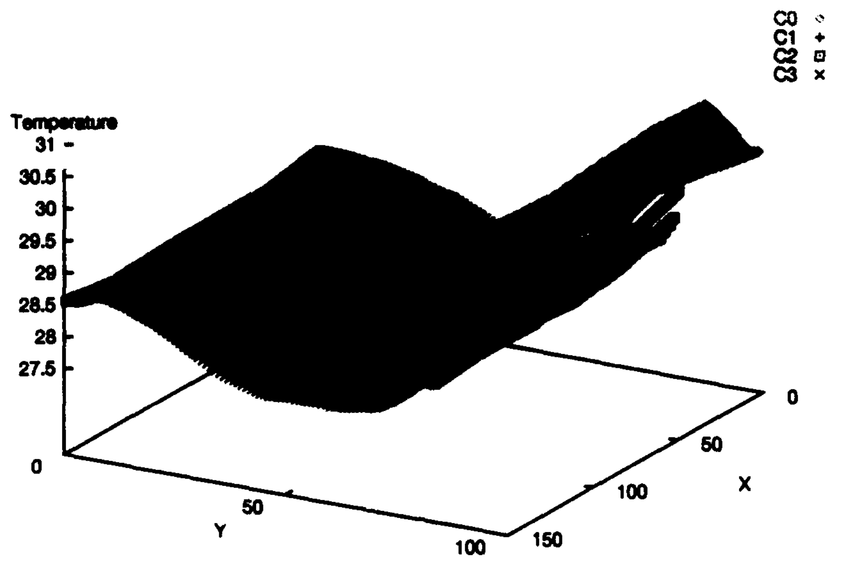

Figure 7: Day 1: Temperature 3D representation

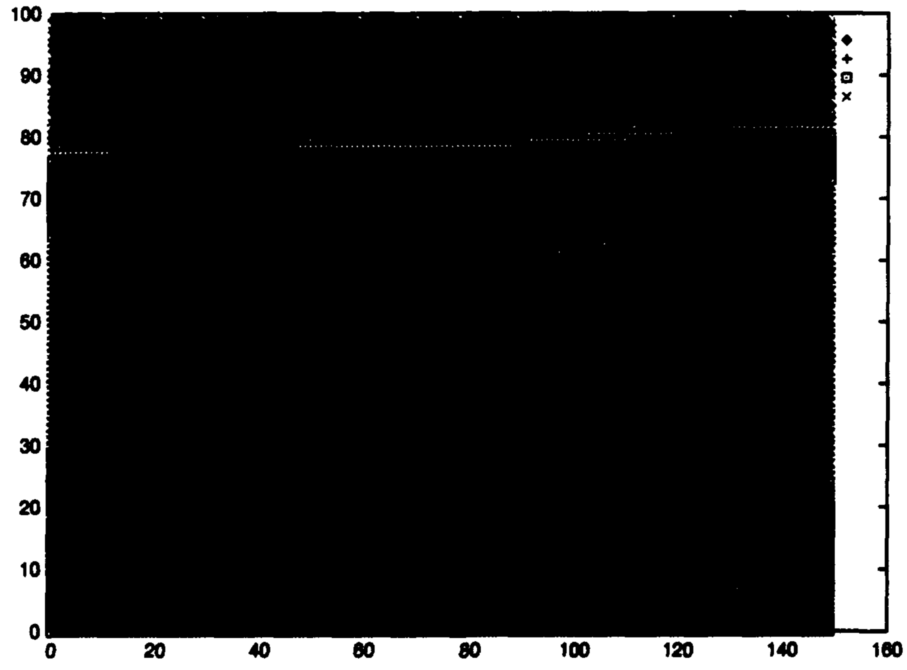

Figure 8: Day 2: Instance space 2D representation

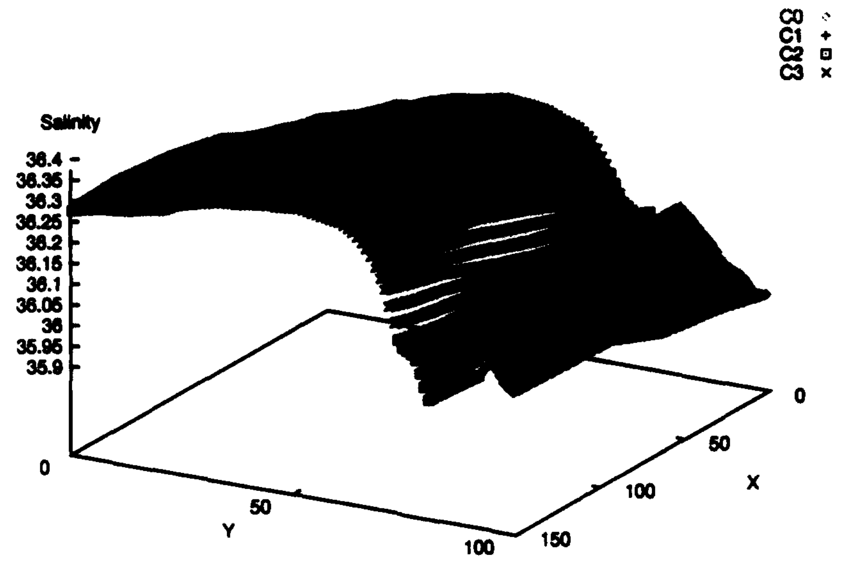

Figure 9: Day 2: Salinity 3D representation

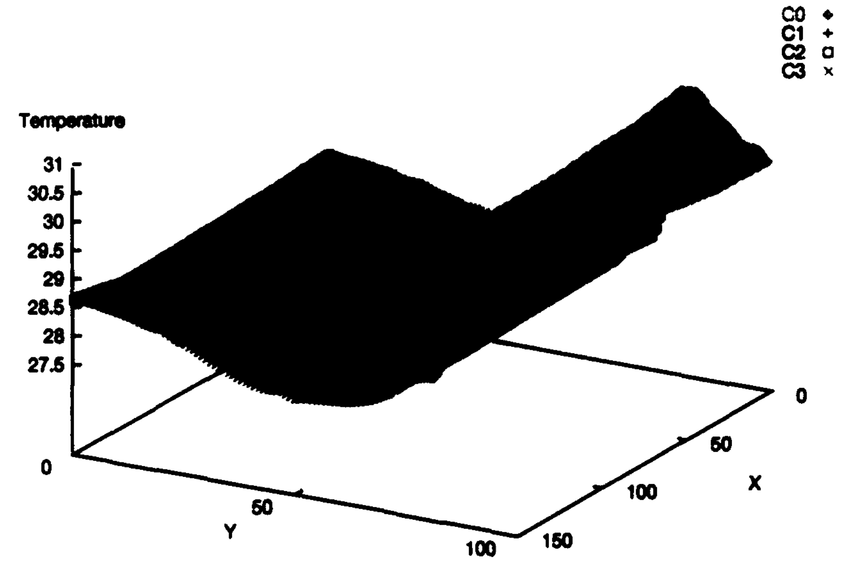

Figure 10: Day 2: Temperature 3D representation

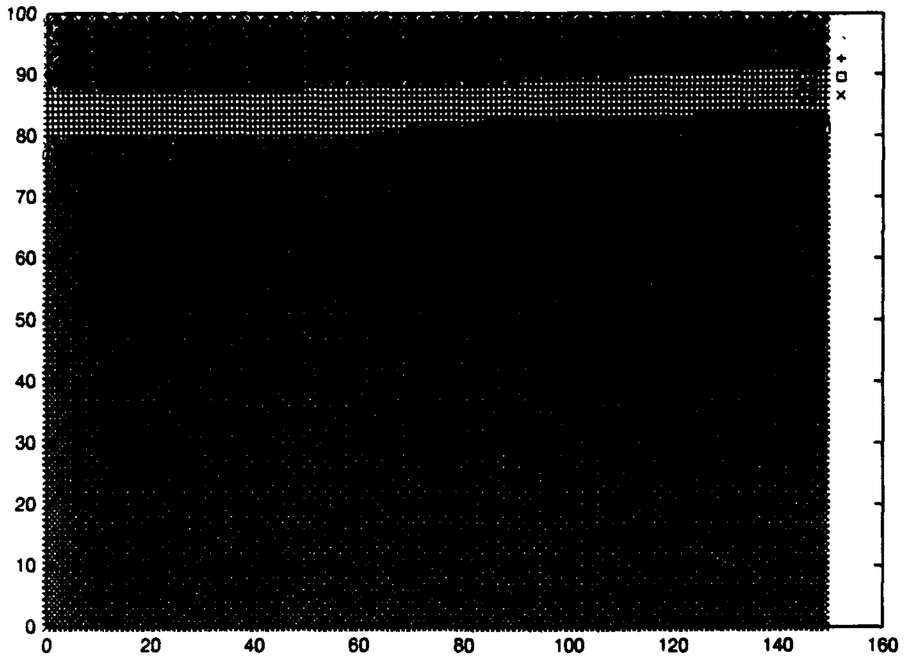

Figure 11: Day 3: Instance space 2D representation

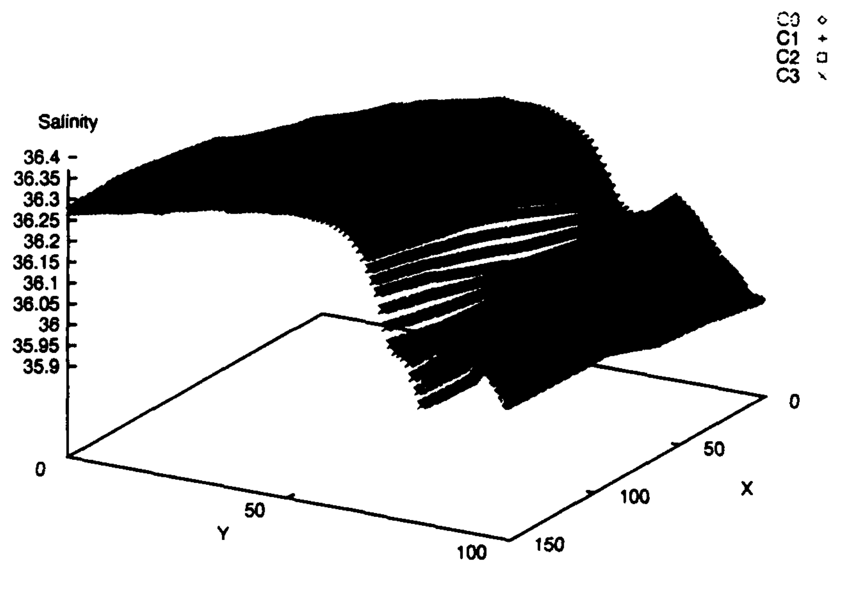

Figure 12: Day 3: Salinity 3D representation

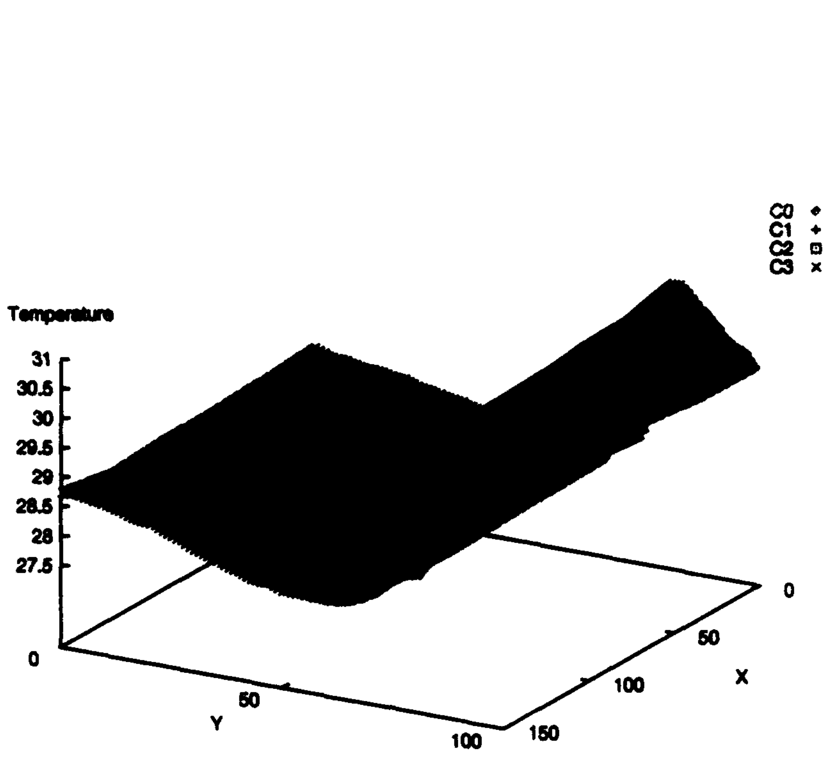

Figure 13: Day 3: Temperature 3D representation

2.5.3 Conclusion.

These experiments show that from raw observational data contained in the scientific databases, it was possible to extract knowledge or useful information hidden in the data. We found correlations between salinity, temperature, and depth as shown in the generated production rules. This allows one to define some features of the Gulf Stream.

We are currently experimenting the learning behavior of ADAS when the system considers several depths over a period of time (several days). We expect to find promising results identifying the Gulf Stream features. Storing these rules in an expert system and using domain knowledge (heuristics), we expect the system to be able to predict Gulf Stream features after learning prior features over a period of time.

3 Interactive Natural Language Interface

The ability to communicate with databases in a natural and convenient way for the human users (i.e., using a natural language, such as English) is of great interest for very large scientific databases. Extending the idea of a system capable to only handle and answer single query, we are convinced that handling a conversation between the human user and the system will enable the machine to better understand and refine the user's intended goals or needs. By conversation, we mean an interactive and intelligent domain specific dialogue man-machine. Using a natural language system will allow to explain clearly the findings discovered in the scientific databases, ambiguities in the query formulation can be resolve, the user phrases the query in an incremental manner (i.e., query refinement), the system recognizes query intended by the user when they are incomplete, and it is application independent.

This section discusses the first phase in the design and development of ENTRETIEN, a computational simulation model of domain specific conversation or discourse in natural language. Note that the concept of conversation is still not well understood and modelized neither in artificial intelligence, computational linguistics, nor cognitive psychology. ENTRETIEN will attempt to explain how a domain specific conversation is handled and used by implementing its processing. It will serve as a basis computational model for effective and efficient human-computer interactions. It is therefore the essential element of the interactive natural language conversational interface of the Intelligent Support System (ISS) for knowledge discovery in environmental databases. Note that the present study only considers the design of domain specific models, not general devices. Note also that changing the domain knowledge of the system makes it able to converse in various and different domains, or databases.

There are mainly three processes involved in handling a natural language dialogue, as depicted by Figure 14. The first process is to understand the user's input utterance (i.e., analysis phase), and to map it into an internal representation that captures its intended meaning. Second is the reasoning and planning process which decides on the form and content of the conversation. For example, it combines the current input utterance in the context of the dialogue (i.e., recognition and synthesis of discourse), checks whether knowledge exists in the knowledge base that may satisfy the user's needs, decides how to represent and accumulate information, and decides whether and what to communicate next to the user. Third is the synthesis process that generates answers to the user. Thus, the issues of knowledge representation, understanding and generating natural language utterances, how the system modifies its knowledge and action by making plans, and the reasoning process will be explained.

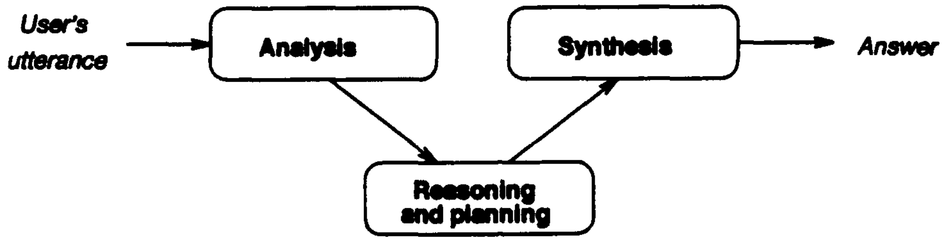

Figure 14: Cognitive tasks involved in a dialogue

The development of the model of conversation has been approached by the following three steps:

1. In step 1, we examined in depth the process of natural language understanding and processing, including the process of conversation in terms of artificial intelligence, computational linguistics, and cognitive psychology. The natural language is English. The better the model is similar to what is actually known about the human cognitive processes involved during a dialogue, the better it will be able to understand and communicate with human users. We extracted from this study the important characteristics the model should possess.

2. In step 2, we surveyed and studied existing models that deal with natural language understanding and processing. This was done to decide whether to use one of them, a combination, or to design a new one, to agree with the necessary characteristics determined at phase 1. We also decided what to represent, and how to represent it.

3. Step 3 is the actual phase of the present research, in which we are developing ENTRETIEN, a conversation model to be implemented for ISS. The model will be developed in two phases:

    - Phase 1 is the design and developement of a model of language for single utterance. The analysis and synthesis systems to be designed in this phase process natural language sentences. The analysis process decodes (i.e., parses) the user ideas and needs expressed in English into an internal representation manageable by the intelligent system. Natural language processing (NLP) makes the machine understand a subset of natural language by breaking up and analyzing the intended meaning of the input sentence (user's question, etc.). It involves four levels of analysis: morphological, syntactical, semantics, and pragmatics. The purpose of any language is to enable communication of ideas and needs. In other words, it is a means for a speaker who has something in mind that he wishes the recipient to understand or be aware of. The synthesis process is merely the reverse of the analysis process and is used to generate answers to the user in a natural way.
    - Phase 2 is the design and development of the model of conversation per se. A conversation is a two-way process of exchange of information by interrelated sentences. It involves at least two

participants. The model of conversation is a more general model than the previous one designed in phase 1. In fact, it contains the model of language of phase 1 (i.e., some knowledge about the language common to both participants) in addition to some knowledge of the surrounding world (i.e., a representation of the domain knowledge), some knowledge about the other participant (a model of what the participant knows, believes, desires, and intends), and the understanding and representation of the context in which the conversation occurs. In this phase, the system becomes a participant in an interactive and intelligent domain specific dialogue with the human user, making an evolution from the previous state (phase 1) in where it was only a simple passive listener. This has the advantages to better define and understand the user's needs in querying the knowledge base and to explain to the user the discoveries contained in the knowledge base and thus not explicitly understandable for human. The model will include the main characteristics of a dialogue such as to infer meaning and to find interrelation between successive sentences (questions, etc.) of the same conversation, to determine whether and when to take the initiative in the dialogue, etc.

3.1 Processing single utterance.

Processing single utterance implies understanding the input sentence (analysis phase) and generating an answer to the user (synthesis phase). The natural language analysis phase breaks up and analyzes a given sentence to determine its intended meaning. This is done in three analysis steps: morphological, syntactic, and semantic analysis. Note that they may not be performed in that order, but rather each process is used as needed. This is depicted by the functional diagram given in Figure 15. To understand dialogue, the system needs to have the following kinds of knowledge:

- *Morphemics knowledge* contains the rules that describe how units of meaning (morphemes) are put together to form words. It encodes rules for plural, verb conjugation, etc.
- *Syntactic knowledge* contains the rules of the grammar that describe the sentence formation. It puts constraints on the number of legal sentences.
- *Semantics knowledge* contains the meaning of words and sentences, hence it puts constraints on the sentence understanding.

Our computational simulation model of language processing will contain the basis for the analysis and the synthesis processes. The scope and precision of the language will enhance and grow with experience.

3.1.1 Representing knowledge with case frames.

Our approach used case frames to represent the knowledge about the surrounding world. Originally developed by Fillmore [Fil68], it is still under active research and widely used [CH92]. It is a means to organize efficiently large amounts of knowledge used for cognitive tasks [BF81]. The fact that it is recursive by nature, and that it combines "bottom-up recognition of key constituent" (i.e., solve problem of goal direction) "with top-down instantiation of less structured constituents" [CH92], makes it very attractive for computational simulation of natural language processing.

A case frame contains a *head concept* (a verb) and a set of *semantic roles* (or cases). Roles may be optional and may contain default values; this corresponds to the case when the system infer expectation on the meaning of the actual situation. A frame contains slots to be filled according to the information provided by the dialogue. Slots correspond to the type of information that must be found in the conversation to be understandable. Links between frames represent additional knowledge required by the context of the current situation. Frames are dynamic entities, and thus may be expanded on the basis of new experience throughout the dialogue.

Case frame is a convenient way to combine both declarative and procedural knowledge. *Declarative knowledge* is a standard data structure which contains objects, properties of the objects, and relations among objects. This corresponds to the declarative form of the frame in which slots are filled according to

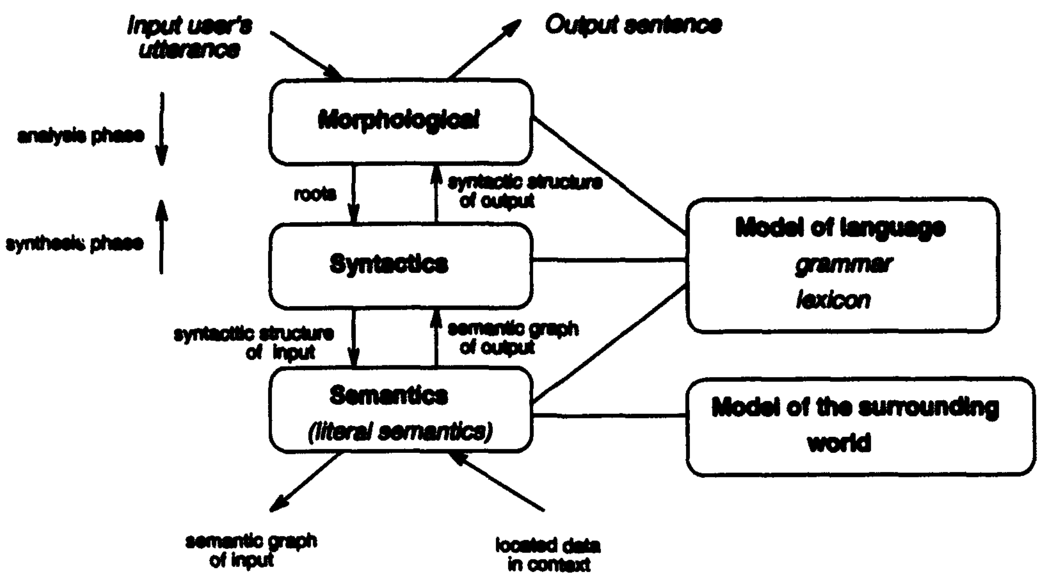

Figure 15: Analysis and synthesis process

information inferred from the input sentence. In this case, knowledge is accessed using descriptions of the data. *Procedural knowledge* is not explicitly stated. It corresponds to knowledge used when executing a function. This is implemented by allowing slots to functions used only if the slots is in use. While declarative knowledge is more efficient, easily accessed and modified, procedural knowledge allows more flexibility.

3.1.2 Case grammar theory.

Case theory is an efficient formalism to represent sentence meaning from its concepts. The case structure has been traditionally studied in linguistics. The new interest for case grammar is due to Fillmore [Fil68] who intended to generalize the transformational generative grammar theory. We follow Fillmore deep structure definitions of case grammar as the basis for our system [Fil68] [Fil71]. Deep case is mainly a classification of the noun phrases present in a sentence, and is in accordance with their conceptual roles in the action described by the sentence [CH92]. Note that only a small finite set of conceptual roles is necessary to represent the meaning of any sentence [BM92] [CH92].

Cases . Fillmore defined the following set of semantic roles (from [BM92] and [Fil71]):

Agent	The investigator of the event, an animate being
Counter agent	The force or resistance against which the action is carried out
Object	The entity that moves or changes or whose position or existence is in consideration
Result	The entity that comes into existence as a result of the event
Instrument	The stimulus or immediate physical cause of an event
Source	The place from which something moves
Goal	The place to which something moves
Experience	The entity that recieves or accepts or experiences or undergoes the effect of an event

Case frames. Each verb of the language is represented by a case frame. This represents the expectation of the event denoted by that verb. Cases may be optional, required, or not needed. *A case frame consists of* two components [CH92]:

- **Case structure**: sets of cases that play a role in the event described by the verb of the phrase. Each slot consists of both a filler and a lexical marker.
- **Selection restriction**: semantic constraints on the choice of relevant cases. For example, agent may be required to be an animate object to consider that case, etc.

Case grammar definition. The following defines (after Fillmore) the case grammar rules:

$S \Rightarrow M + P$
$M \Rightarrow$all modess
$P \Rightarrow V + C_1 ... C_n$
$V \Rightarrow$all verbs of the language
$C_i \Rightarrow K + NP$
$K \Rightarrow (PREP)$
$NP \Rightarrow (PREP) + (DET) + (ADJ|N) * +N + (S|NP)$

A sentence is mainly composed of a modality M and a proposition P. P is a verb V and a set of cases C_i. Each C_i is a Kasus K and a noun phrase NP. K is a preposition introducing a noun phrase; it may not be present.

Modality. The mode is a general description of a sentence. Simmons [Sim73] defined the following mode values (from [Har85]):

Mode	Description	Values
Tense	Indicate time orientation	present, pas, future
Aspect	Say whether the event is continuing	perfect, imperfect
Form	Information about the verb phrase	simple, emphatic, progressive
Mood	Relates to the elements order	declarative, interrogative, imperative
Essence	Say whether verb phrase is positive	positive, negative, indeterminate
Modal	Helping verb adding features to main verb	may, can, must
Manner	Indicated by adverbial part of senetence	adverbial
Time	Indicated by adverbial part of senetence	adverbial

3.1.3 Parsing with case frames.

During the analysis process, parsing, cases of a frame must be mapped to the noun phrase. For the synthesis phase (sentence generation), the concept to express is mapped into the case frame of interest. In both cases, whether a slot is ooptional, required, or not needed decides on what to map. The lexicon is used to access the word semantics knowledge to decide which case to fill in.

Case frame parsing algorithm. Carbonell and Hayes [CH92] proposes the following algorithm for parsing with case frames.

- Browse all case frame to match input. If there exist some candidate(s) then continue to next step, else stop since the parser cannot parse that input.
- Attempt to recognize in the input sentence each case of the selected case frame.
  1. If case is marked lexically, do a match for the case marker.
  2. If the case is marked positionally, do a match of case filler.
  3. If case could be either way, search first for lexical marker, then positional marker if failed.
- Attempt to recognize all optional cases.

3.1.4 Lexicon.

The dictionary is a data structure storing knowledge about the language. Chunk of knowledge are represented with frames and scripts. The lexicon is mainly used during the parsing process to access syntactic and semantic information about the words of the language. Note that it gives only meaning of words considered individually. The lexicon also stores the case frames for each verb of the language as discussed previously.

For storage space purposes, the lexicon only stores the roots of the words. Related words formed on those roots are determined from the information contained in each entry word (whether and which affixes are accepted) and during the morphological processing. This also saves search time.

Implementation. We intend to use the expert system shell CLIPS 6.0 to develop the lexicon. The choice is justified by the excellent Rete algorithm implemented in CLIPS, which is efficient in organizing and seraching for dictionary entries.

3.2 Processing a conversation.

This section describes the design of a computational simulation model of conversation, based on human cognitive models, to conduct intelligent and interactive dialogues with the user to better exploit information from the knowledge base and to explain in a natural way findings that have been discovered in the scientific databases. A conversation, or dialogue, is an extremely complex process [Pop86] [Har85] [ErH93] that is still not well understood and modelized in artificial intelligence, computational linguistics, and cognitive psychology. The following proposes an approach in designing such a model of conversation, limited to discourses. Discourses are domain specific conversations. A conversation is an interactive two-way process of exchange of information by a succession of interrelated sentences. It involves at least two participants, the present case being limited to two participants: a human user and the machine. The machine will interpret and refine the user's needs in an interactive manner.

3.2.1 General approach.

The model is based on the model of the language discussed in the previous section to which are added two more important features (cf. Figure 16):

- *Pragmatics knowledge* contains the rules for the conversation.
- *Reasoning and planning process.*

3.2.2 Main characteristics.

The process of conversation should have the following main characteristics to interpret the meaning of the dialogue.

- *Tacit knowledge.* The model should perceive more information that what is given by the sentence. Hence, it should get tacit knowledge (i.e., presuppositions), and connection between sentence (i.e., implication and inference). Each sentence conveys additional indirect information that should be transformed as explicit information. This may be done by analyzing every word, its use and relation to other words in the sentence, as well as previous knowledge acquired from prior sentences since the beginning of the conversation.
- *Approximate answers.* When a participant X converses with another participant Y (via questions, etc.), X anticipates Y's answer. In the case that Y does not know, or is not sure about the answer, Y generates an approximate or indirect answer [Pop86]. Human always generate such answers, thus the model should be able to handle the type of answer that a human user may generate. Approximate answers are used when a precise answer to the given question gives no information, the answer is not known, and when the system ask to clarify words it did not understand.

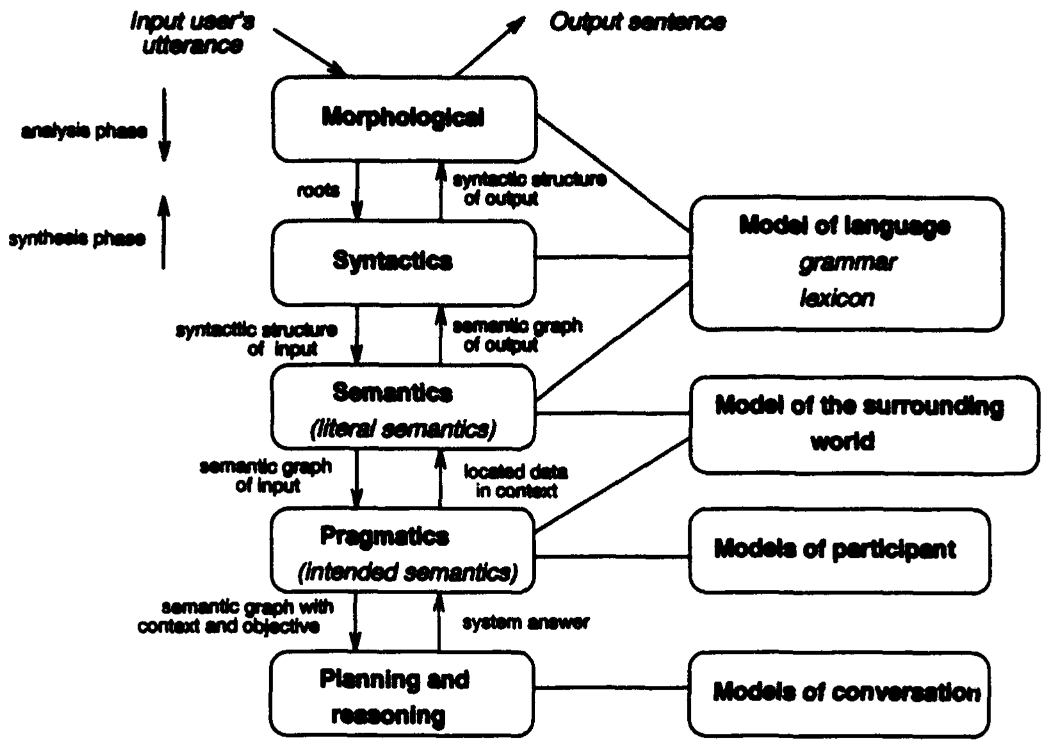

Figure 16: Case frame data structure

- *Unambiguous statements.* Statements generated by a participant should be unambiguous in terms of syntax and semantics. Each participant should have knowledge about:
  - a common language to conduct a conversation (i.e., English).
  - the surrounding world, i.e., the domain knowledge, such as oceanography.
  - the context in which the conversation occurs.
  - the other participant to help infer and deduce indirect sentences, intended meanings, etc.
- *Anaphoric references.* To handle a dialogue, the model should be able to recall references to previous sentences to understand the previous sentence.
- *Ellipsis.* The model should be able to determine the meaning of slightly incomplete sentences.
- *Initiative in the conversation.* When needed, the system should take the initiative in the conversation, and not be only a passive question answering machine. This is done to clarify points, to refine the user's needs, etc.
- *Synonyms and homonyms handling.* The representation, in terms of meaning, of synonym phrases should be identical, in order to deduce the same meaning.
- *Depth of understanding.*

3.2.3 Computational model of the user's plans.

In a natural language information-seeking dialogue, human beings infer plans and goals of the speaker. Information seeking dialogue is a complex two-way process that involves two participants, in which the information-seeker (IS) is the human user and the information-provider (IP) is the computer system. We want to base our system on a previous simulation model developed by [Car90] which seems the most appropriate actual system for further enhancements towards a more robust system that will be a step closer to the human way of handling dialogues. Our goal is to palliate an obvious lack of efficient and effective communication between the human user seeking information and the computer system willing to provide such information. Human cognition can provide an inspiring model for the design of knowledge systems. This is a very on-going complex task that many psychologists, linguists, and artificial intelligence computer scientists are trying to accomplish for several years. We focused on the *plan recognition* aspect of the model, and more specifically on how to relate a new utterance to the actual model built from previous utterances.

IS recognition of plan. A conversation, or dialogue, is a planned activity. Plans and goals represent the participant's actual beliefs and expectations that are used to make the next statement in the conversation (e.g., answer to a question, transfer of information from one person to another, etc.) [Car88]. During a dialogue, the system (IP) dynamically analyzes each utterance to understand the user (IS) underlying intentions to generate better answers. This is done by searching a space of plausible goals (i.e., partial plans) related to the actual dialogue context. In fact, the system tries to guess the most plausible goal to represent the actual IS model. To understand a discourse requires one to infer the intentions and beliefs of IS. This requires one to derive a plan with a sequence of actions to achieve. This idea of making strategies to satisfy a goal has been exploited for decades (e.g., [Est78][And80]). Allen and Perrault [AP80] developed a system that derives intentions behind surface speech acts [Sea70], using knowledge about the plans and beliefs about the speaker's goals. In his work, plan inference is determine by first order predicate logic rules and control strategy. Searle's approach to formulate speech acts was based on the idea that human beings plan the generation of utterance in natural language dialogue, so we must do a model that simulates human behavior in language comprehension. This idea was later formalized by Allen as stated above [All83] [AP80]and also by Cohen and Perrault [CP79]. The set of possible plans are represented as a hierarchical structure, a tree, in which like the human memory is an organization with a top-down recall scheme. The more the plan is near a leaf, the less likely it will be considered next since the less easier to retrieve from the structure (i.e., it takes more effort to retrieve deep nodes). Thorndyke was the first to address the problem of memory representation of connected utterances [Tho77].

Plan recognition in dialogue. The main issue is the focus of attention in the dialogue. Focus is the process of main concentration of one participant on the other participant knowledge base [Gro81]. This is related to the actual discourse context. Due to the large size of one's knowledge base, they rather concentrate on the actual space of the knowledge base that agrees with the dialogue current context. McKeown further developed Grosz theory by claiming that when a speaker has the choice of the topic of the discussion, this speaker always chooses the most recently introduced topic, if he has still something to say about it [McK83]. This is the basis of Carberry's theory on the natural organization of the human dialogue [Car90].

Objectives. We would like to develop a robust computational dynamic model of the information-seeker (IS) plans and goals during an information-seeking type natural dialogue. To do so, we would like to improve an existing model developed by Carberry [Car90] (that seems most promising for enhancements, by relaxing actual restrictive constraints) towards a more robust system of the second generation of plan recognition systems. The issues of the present study refer then to several fields, and therefore the criteria of success of the cognitive simulation model will be quite different for each field [Kie92]. For Artificial Intelligence, in contributing towards the construction of more intelligent machines, and more specifically to render machines able to communicate in natural language with their human users; For Cognitive Psychology, in attempting to advance explanations on some cognitive processes involved during dialogue, such as utterance understanding and processing, comprehension, inference, knowledge representation in memory for information retrieval, and human problem solving and planning; For Computational Linguistics, in implementing theories on natural language that implies to specify details necessary for a computer to handle, but not necessary evident in complex theories. Hence, we will center this study among the above fields, in reverse to the general tendency to conduct studies in the boundaries of a single field.

4 Future Directions

This section proposes the future directions we considered to follow in designing and implementing the complete system (Figure 1).

4.1 Determining the number of clusters.

Recall that the data clustering process produces a hierarchical tree in where instances are classified according to some distance measure. Deciding where to cut the tree to have the best number of clusters is an issue not yet well formalised in statistics. In the future, we plan to follow an approach inspired by Romesburg [Rom84], and described in [PP94], that seems to be most similar to the way human beings select the number of clusters during taxonomic cognitive processes.

4.2 Data clustering algorithms.

We intend to use more computationally efficient data clustering algorithms, such those contained in the ClusPak package developed by Besdek and Pal [BP92b] [BP92a]. Future tasks include to develop an interface between the system and this package.

4.3 Incremental Updating of the Knowledge Base System Design.

The system for incrementally updating the knowledge base (IUKBS) is intended to combine every new set of production rules that describe the new data inserted in the databases with the previous rules contained in the knowledge base of an expert system. The intent is to keep pace with the changes that occur in the data, since large scientific databases are dynamic entities. The issues in updating incrementally the knowledge base involve checking for rules consistency, i.e., to carefully manage the database of production rules to update accordingly every other rule that is modified, either directly or indirectly, each time a new rule is added or removed.

4.4 Use of an expert system shell.

Since the main goal is to use the knowledge base for information retrieval (answering user's queries) and to infer new knowledge from the data, it is of interest to transform the production rules generated by Quinlan's C4.5rules into CLIPS format. CLIPS is an expert system shell developed by NASA [NAS91]. This will allow us to combine them with the domain knowledge (environmental definitions, etc.) already captured in a frame-based semantic network implemented using CLIPS during a previous work [PP94].

4.5 Implementing ENTRETIEN.

This concerns implementing every process discussed in the natural language interface system (cf. Section 3).

5 Conclusion

This research clearly shows that combining machine learning techniques (data clustering and induction learning by decision trees) with production rules is an effective mean for automatic data analysis of huge and rapidly changing amounts of observational raw data in scientific databases. In terms of expert systems, automatic rule generation from non-trivial knowledge contained in scientific databases eases the knowledge acquisition bottleneck that usually exists in extracting knowledge from human experts. Further developments of the prototype seems promising in solving the problem of automatic data analysis for information retrieval in scientific databases.

We believe that constructing the natural language interface, described in this report, in which the computer "speaks" the same language as the human user, is of great interest to efficiently exploit information non-trivially stored in databases. We want to go a step further in building an intelligent machine capable of communicating efficiently, as human do naturally. Language is the support of information transfer. Communication is then the underlying support of intelligence.

6 Acknowledgements

We would like to thank Bill Thomason for his helpful contribution in using the C4.5 package, Don Goff for his precious help in satisfying our computational needs, Dr. Lanny Yeske for reviewing this report, and all friendly CAST researchers to whom we appreciate the advice, support, and collaboration.

This research was supported by the Office of Naval Research's Navy Ocean Modeling and Prediction Program under Research Grant N00014-92-J-4109 with Mississippi State University.

7 Appendices

7.1 Gain criterion

The notion of $gain(X)$ mentioned in the section about the decision tree refers to the $gain(X)$ function used in the gain criterion.

After T has been partitioned with respect to the n outcomes of a test X, the expected information requirement can be found as the weighted sum over the subsets as $info_x(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \times info(T)$

Essentially, $info(T)$ measures the avery amount of information needed to identify the class of a case in T. $info(T) = -\sum_{j=1}^{k} \frac{freq(C_j,T)}{|T|} \times \log_2 \left(\frac{freq(C_j,T)}{|T|}\right)$

Therefore,$gain(X) = info(T) - info_x(T)$measures the information gained by partitioning T according to the test X.

7.2 Day 1: Decision tree obtained for the clusters' description

```
Read 15000 cases (6 attributes) from ts1.data
Decision Tree:
temperature <= 29.0741 :
|   temperature <= 28.074 :
|   |   salinity <= 36.1538 : C2 (25.0/1.0)
|   |   salinity > 36.1538 :
|   |   |   salinity > 36.1599 : C3 (5892.0/1.0)
|   |   |   salinity <= 36.1599 :
|   |   |   |   temperature <= 28.0632 : C3 (17.0)
|   |   |   |   temperature > 28.0632 : C2 (8.0)
|   temperature > 28.074 :
|   |   temperature <= 28.0792 :
|   |   |   salinity <= 36.2189 : C2 (22.0)
|   |   |   salinity > 36.2189 : C3 (20.0)
|   |   temperature > 28.0792 :
|   |   |   temperature <= 28.0917 :
|   |   |   |   salinity <= 36.2798 : C2 (61.0)
|   |   |   |   salinity > 36.2798 :
|   |   |   |   |   salinity > 36.3015 : C3 (19.0)
|   |   |   |   |   salinity <= 36.3015 :
|   |   |   |   |   |   temperature <= 28.087 : C3 (6.0)
|   |   |   |   |   |   temperature > 28.087 :
|   |   |   |   |   |   |   salinity <= 36.2903 : C3 (2.0)
|   |   |   |   |   |   |   salinity > 36.2903 : C2 (3.0)
|   |   |   temperature > 28.0917 :
|   |   |   |   temperature > 28.0948 : C2 (6496.0/1.0)
|   |   |   |   temperature <= 28.0948 :
|   |   |   |   |   column > 34 : C2 (14.0)
|   |   |   |   |   column <= 34 :
|   |   |   |   |   |   salinity > 36.3185 : C2 (5.0)
|   |   |   |   |   |   salinity <= 36.3185 :
|   |   |   |   |   |   |   temperature <= 28.0936 : C3 (3.0/1.0)
|   |   |   |   |   |   |   temperature > 28.0936 : C2 (3.0)
temperature > 29.0741 :
|   temperature <= 29.8921 : C1 (252.0/1.0)
|   temperature > 29.8921 :
|   |   temperature > 29.9348 : C0 (2121.0)
|   |   temperature <= 29.9348 :
|   |   |   column <= 85 : C1 (9.0)
|   |   |   column > 85 : C0 (22.0)

Simplified Decision Tree:
temperature <= 29.0741 :
|   temperature <= 28.074 :
|   |   salinity <= 36.1538 : C2 (25.0/2.5)
|   |   salinity > 36.1538 :
|   |   |   salinity > 36.1599 : C3 (5892.0/2.6)
|   |   |   salinity <= 36.1599 :
|   |   |   |   temperature <= 28.0632 : C3 (17.0/1.3)
|   |   |   |   temperature > 28.0632 : C2 (8.0/1.3)
|   temperature > 28.074 :
|   |   temperature <= 28.0792 :
|   |   |   salinity <= 36.2189 : C2 (22.0/1.3)
|   |   |   salinity > 36.2189 : C3 (20.0/1.3)
|   |   temperature > 28.0792 :
|   |   |   temperature > 28.0917 : C2 (6521.0/5.1)
|   |   |   temperature <= 28.0917 :
|   |   |   |   salinity <= 36.2798 : C2 (61.0/1.4)
|   |   |   |   salinity > 36.2798 :
|   |   |   |   |   salinity > 36.3015 : C3 (19.0/1.3)
|   |   |   |   |   salinity <= 36.3015 :
|   |   |   |   |   |   temperature <= 28.087 : C3 (6.0/1.2)
|   |   |   |   |   |   temperature > 28.087 :
|   |   |   |   |   |   |   salinity <= 36.2903 : C3 (2.0/1.0)
|   |   |   |   |   |   |   salinity > 36.2903 : C2 (3.0/1.1)
temperature > 29.0741 :
|   temperature <= 29.8921 : C1 (252.0/2.6)
|   temperature > 29.8921 :
|   |   temperature > 29.9348 : C0 (2121.0/1.4)
|   |   temperature <= 29.9348 :
|   |   |   column <= 85 : C1 (9.0/1.3)
|   |   |   column > 85 : C0 (22.0/1.3)

Evaluation on training data (15000 items):

 Before Pruning           After Pruning
```

```
----------------  -------------------------
Size      Errors  Size      Errors  Estimate
  39    5( 0.0%)    31    6( 0.0%)    ( 0.2%)   <<
```

7.3 Day 1: Production rules generated from the decision tree

```
Read 15000 cases (6 attributes) from ts1
Processing tree 0
Final rules from tree 0:
Rule 19: temperature > 29.9348
 -> class C0  [99.9%]
Rule 18: temperature > 29.8921
      column > 85
 -> class C0  [99.9%]
Rule 17: temperature > 29.0741
      temperature <= 29.9348
      column <= 85
 -> class C1  [99.4%]
Rule 16: temperature > 29.0741
      temperature <= 29.8921
 -> class C1  [99.0%]
Rule 2:  salinity > 36.1538
      temperature <= 28.0632
 -> class C3  [100.0%]
Rule 6:  salinity > 36.2189
      temperature <= 28.0792
 -> class C3  [100.0%]
Rule 8:  salinity > 36.2798
      temperature <= 28.087
 -> class C3  [100.0%]
Rule 11: salinity > 36.3015
      temperature <= 28.0917
 -> class C3  [100.0%]
Rule 4:  salinity > 36.1599
      temperature <= 28.074
 -> class C3  [100.0%]
Rule 7:  salinity <= 36.2798
      temperature > 28.0792
      temperature <= 29.0741
 -> class C2  [100.0%]
Rule 14: temperature > 28.0917
      temperature <= 29.0741
 -> class C2  [99.9%]
Rule 5:  salinity <= 36.2189
      temperature > 28.074
      temperature <= 29.0741
 -> class C2  [99.9%]
Rule 10: salinity <= 36.3015
      temperature > 28.087
      temperature <= 29.0741
 -> class C2  [99.9%]
Rule 3:  salinity <= 36.1599
      temperature > 28.0632
      temperature <= 29.0741
 ->  class C2  [99.9%]
Rule 1:  salinity <= 36.1538
      temperature <= 29.0741
 ->  class C2  [99.8%]
Default class: C2

Evaluation on training data (15000 items):
Rule  Size  Error  Used  Wrong            Advantage
----  ----  -----  ----  -----            ---------
  19     1   0.1%  2121      0 (0.0%)      66 (66|0)  C0
  18     2   0.1%    22      0 (0.0%)      22 (22|0)  C0
  17     3   0.6%   238      0 (0.0%)       9 (9|0)  C1
  16     2   1.0%    23      1 (4.3%)      22 (22|0)  C1
   2     2   0.0%  5851      0 (0.0%)      17 (17|0)  C3
   6     2   0.0%    65      0 (0.0%)       5 (5|0)  C3
   8     2   0.0%    17      0 (0.0%)       6 (6|0)  C3
  11     2   0.0%     8      0 (0.0%)       8 (8|0)  C3
   4     2   0.0%    13      1 (7.7%)      11 (12|1)  C3
   7     3   0.0%  5259      1 (0.0%)       0 (0|0)  C2
  14     2   0.1%  1323      2 (0.2%)       0 (0|0)  C2
   5     3   0.1%    22      0 (0.0%)       0 (0|0)  C2
  10     3   0.1%     5      2 (40.0%)       0 (0|0)  C2
   3     3   0.1%    27      0 (0.0%)       0 (0|0)  C2
   1     2   0.2%     6      1 (16.7%)       0 (0|0)  C2

Tested 15000, errors 8 (0.1%)   <<

  (a)  (b)  (c)  (d) <-classified as
 ---- ---- ---- ----
 2143    1           (a): class C0
       260    1      (b): class C1
           6636    1 (c): class C2
                5 5953 (d): class C3
```

7.4 Day 2: Decision tree obtained for the clusters' description

```
Read 15000 cases (6 attributes) from ts2.data
Decision Tree:
temperature <= 28.9968 :
|   temperature <= 28.1988 :
|   |   temperature <= 28.1669 : C3 (6298.0/1.0)
|   |   temperature > 28.1669 :
|   |   |   salinity <= 36.2868 :
|   |   |   |   salinity > 36.2057 : C2 (67.0)
|   |   |   |   salinity <= 36.2057 :
|   |   |   |   |   salinity > 36.1474 : C3 (9.0)
|   |   |   |   |   salinity <= 36.1474 :
|   |   |   |   |   |   salinity <= 36.1296 : C2 (56.0)
|   |   |   |   |   |   salinity > 36.1296 :
|   |   |   |   |   |   |   temperature <= 28.1838 : C3 (8.0/1.0)
|   |   |   |   |   |   |   temperature > 28.1838 : C2 (8.0)
|   |   |   salinity > 36.2868 :
```

```
|   |   |   |   temperature <= 28.1957 : C3 (107.0)
|   |   |   |   temperature > 28.1957 :
|   |   |   |   |   salinity <= 36.3162 : C2 (3.0)
|   |   |   |   |   salinity > 36.3162 : C3 (10.0)
|   temperature > 28.1988 :
|   |   temperature > 28.2035 : C2 (5227.0)
|   |   temperature <= 28.2035 :
|   |   |   salinity <= 36.318 : C2 (22.0)
|   |   |   salinity > 36.318 : C3 (3.0)
temperature > 28.9968 :
|   temperature <= 29.9177 : C1 (243.0)
|   temperature > 29.9177 :
|   |   temperature > 29.9634 : C0 (2886.0)
|   |   temperature <= 29.9634 :
|   |   |   salinity <= 36.0445 : C0 (34.0/1.0)
|   |   |   salinity > 36.0445 : C1 (19.0)

Simplified Decision Tree:
temperature <= 28.9968 :
|   temperature <= 28.1988 :
|   |   temperature <= 28.1669 : C3 (6298.0/2.6)
|   |   temperature > 28.1669 :
|   |   |   salinity <= 36.2868 :
|   |   |   |   salinity > 36.2057 : C2 (67.0/1.4)
|   |   |   |   salinity <= 36.2057 :
|   |   |   |   |   salinity <= 36.1296 : C2 (56.0/1.4)
|   |   |   |   |   salinity > 36.1296 :
|   |   |   |   |   |   temperature <= 28.1838 : C3 (16.0/2.5)
|   |   |   |   |   |   temperature > 28.1838 : C2 (9.0/2.4)
|   |   |   salinity > 36.2868 :
|   |   |   |   temperature <= 28.1957 : C3 (107.0/1.4)
|   |   |   |   temperature > 28.1957 :
|   |   |   |   |   salinity <= 36.3162 : C2 (3.0/1.1)
|   |   |   |   |   salinity > 36.3162 : C3 (10.0/1.3)
|   temperature > 28.1988 :
|   |   temperature > 28.2035 : C2 (5227.0/1.4)
|   |   temperature <= 28.2035 :
|   |   |   salinity <= 36.318 : C2 (22.0/1.3)
|   |   |   salinity > 36.318 : C3 (3.0/1.1)
temperature > 28.9968 :
|   temperature <= 29.9177 : C1 (243.0/1.4)
|   temperature > 29.9177 :
|   |   temperature > 29.9634 : C0 (2886.0/1.4)
|   |   temperature <= 29.9634 :
|   |   |   salinity <= 36.0445 : C0 (34.0/2.6)
|   |   |   salinity > 36.0445 : C1 (19.0/1.3)


Evaluation on training data (15000 items):

 Before Pruning          After Pruning
----------------  -------------------------
```

```
Size      Errors   Size      Errors   Estimate
  31    3( 0.0%)     29    4( 0.0%)    ( 0.2%)   <<
```

7.5 Day 2: Production rules generated from the decision tree

```
Read 15000 cases (6 attributes) from ts2
Processing tree 0
Final rules from tree 0:
Rule 13: temperature > 28.9968
      temperature <= 29.9177
 -> class C1  [99.4%]
Rule 15: salinity > 36.0445
      temperature > 28.9968
      temperature <= 29.9634
 -> class C1  [98.6%]
Rule 16: temperature > 29.9634
 -> class C0  [100.0%]
Rule 14: salinity <= 36.0445
      temperature > 29.9177
 -> class C0  [99.8%]
Rule 12: temperature > 28.2035
      temperature <= 28.9968
 -> class C2  [100.0%]
Rule 10: salinity <= 36.318
      temperature > 28.1988
      temperature <= 28.9968
 -> class C2  [100.0%]
Rule 8:  salinity <= 36.3162
      temperature > 28.1957
      temperature <= 28.9968
 -> class C2  [100.0%]
Rule 6:  salinity > 36.2057
      salinity <= 36.2868
      temperature > 28.1669
 -> class C2  [99.9%]
Rule 4:  salinity <= 36.2868
      temperature > 28.1838
      temperature <= 28.9968
 -> class C2  [99.9%]
Rule 2:  salinity <= 36.1296
      temperature > 28.1669
      temperature <= 28.9968
 -> class C2  [99.9%]
Rule 7:  salinity > 36.2868
      temperature <= 28.1957
 -> class C3  [100.0%]
Rule 11:  salinity > 36.318
      temperature <= 28.2035
 -> class C3  [100.0%]
Rule 1:  temperature <= 28.1669
 -> class C3  [100.0%]
Rule 5:  salinity > 36.1474
      salinity <= 36.2057
```

```
        temperature <= 28.1988
    ->  class C3  [99.6%]
Default class: C3

Evaluation on training data (15000 items):

Rule  Size  Error  Used  Wrong            Advantage
----  ----  -----  ----  -----            ---------
  13     2   0.6%   243      0 (0.0%)     162 (162|0)  C1
  15     3   1.4%    19      0 (0.0%)      19 (19|0)  C1
  16     1   0.0%  2886      0 (0.0%)    1607 (1607|0) C0
  14     2   0.2%    34      1 (2.9%)      33 (33|0)  C0
  12     2   0.0%  5227      0 (0.0%)     124 (124|0)  C2
  10     3   0.0%    22      0 (0.0%)       2 (2|0)  C2
   8     3   0.0%     8      0 (0.0%)       3 (3|0)  C2
   6     3   0.1%    67      0 (0.0%)      31 (31|0)  C2
   4     3   0.1%    31      1 (3.2%)       5 (6|1)  C2
   2     3   0.1%    29      0 (0.0%)      29 (29|0)  C2
   7     2   0.0%  4844      0 (0.0%)       0 (0|0)  C3
  11     2   0.0%    13      0 (0.0%)       0 (0|0)  C3
   1     1   0.0%  1561      1 (0.1%)       0 (0|0)  C3
   5     3   0.4%     8      0 (0.0%)       0 (0|0)  C3

Tested 15000, errors 4 (0.0%)   <<

  (a)  (b)  (c)  (d) <-classified as
 ---- ---- ---- ----
 2919                (a): class C0
    1  262           (b): class C1
           5383    2 (c): class C2
                1 6432 (d): class C3
```

7.6 Day 3: Decision tree obtained for the clusters' description

```
Read 15000 cases (6 attributes) from ts3.data
Decision Tree:
temperature <= 29.5616 :
|   temperature > 28.9764 : C2 (168.0)
|   temperature <= 28.9764 :
|   |   temperature <= 28.8758 : C3 (11782.0)
|   |   temperature > 28.8758 :
|   |   |   salinity <= 36.0483 : C3 (23.0/1.0)
|   |   |   salinity > 36.0483 : C2 (2.0)
temperature > 29.5616 :
|   temperature <= 30.5747 :
|   |   temperature <= 30.5622 : C0 (1839.0)
|   |   temperature > 30.5622 :
|   |   |   salinity <= 36.0993 : C0 (17.0)
|   |   |   salinity > 36.0993 : C1 (8.0/1.0)
|   temperature > 30.5747 :
|   |   temperature <= 30.5912 :
|   |   |   salinity <= 36.0871 : C0 (11.0)
|   |   |   salinity > 36.0871 : C1 (16.0)
|   |   temperature > 30.5912 :
|   |   |   temperature > 30.6055 : C1 (1109.0)
|   |   |   temperature <= 30.6055 :
|   |   |   |   salinity <= 36.0636 : C0 (2.0)
|   |   |   |   salinity > 36.0636 : C1 (23.0)

Evaluation on training data (15000 items):

 Before Pruning           After Pruning
----------------   --------------------------
Size      Errors   Size      Errors   Estimate
  23    2( 0.0%)     23    2( 0.0%)    ( 0.1%)   <<
```

7.7 Day 3: Production rules generated from the decision tree

```
Read 15000 cases (6 attributes) from ts3
Processing tree 0
Final rules from tree 0:
Rule 5:  temperature > 29.5616
      temperature <= 30.5622
 -> class C0  [99.9%]
Rule 8:  salinity <= 36.0871
      temperature > 29.5616
      temperature <= 30.5912
 -> class C0  [99.9%]
Rule 10: salinity <= 36.0636
      temperature > 29.5616
      temperature <= 30.6055
 -> class C0  [99.9%]
Rule 4:  temperature > 28.9764
      temperature <= 29.5616
 -> class C2  [99.2%]
Rule 3:  salinity > 36.0483
      temperature > 28.8758
      temperature <= 29.5616
 -> class C2  [96.9%]
Rule 12:  temperature > 30.6055
 -> class C1  [99.9%]
Rule 11: salinity > 36.0636
      temperature > 30.5912
 -> class C1  [99.9%]
Rule 9:  salinity > 36.0871
      temperature > 30.5747
 -> class C1  [99.9%]
Rule 7:  salinity > 36.0993
      temperature > 30.5622
 -> class C1  [99.7%]
Rule 2:  temperature <= 28.9764
 -> class C3  [100.0%]
Default class: C0

Evaluation on training data (15000 items):
```

```
Rule  Size  Error  Used  Wrong              Advantage
----  ----  -----  ----  -----              ---------
   5     2   0.1%  1839      0 (0.0%)        0 (0|0)  C0
   8     3   0.1%    27      0 (0.0%)        0 (0|0)  C0
  10     3   0.1%     2      0 (0.0%)        0 (0|0)  C0
   4     2   0.8%   168      0 (0.0%)      126 (126|0)  C2
   3     3   3.1%     2      0 (0.0%)        2 (2|0)  C2
  12     1   0.1%  1109      0 (0.0%)       53 (53|0)  C1
  11     2   0.1%    23      0 (0.0%)        4 (4|0)  C1
   9     2   0.1%    16      0 (0.0%)        2 (2|0)  C1
   7     2   0.3%     8      1 (12.5%)       6 (7|1)  C1
   2     1   0.0% 11805      1 (0.0%)    11804 (11804|0)  C3

Tested 15000, errors 2 (0.0%)   <<

  (a)  (b)  (c)  (d)  <-classified as
 ---- ---- ---- ----
 1869    1              (a): class C0
      1155              (b): class C1
            170     1   (c): class C2
                11804   (d): class C3
```

References

[All83] James F. Allen. Recognizing intentions from natural language utterances. In M. Brady and R. C. Berwick, editors, *Computational models of discourse*, pages 107–166, Cambridge, MA, 1983. The MIT Press.

[And80] J. R. Anderson. *Cognitive psychology and its implications*. W. H. Freeman, San Francisco, CA, 1980.

[AP80] James F. Allen and C. R. Perrault. Analyzing intention in utterances. *Artificial Intelligence*, 15:143–178, 1980.

[BF81] Avron Barr and Edward A. Feigenbaum, editors. *The handbook of Artificial Intelligence*, volume 1. Addison-Wesley, Readings, MA, 1981.

[BM92] B. Bruce and M. G. Moser. Grammar case. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 563–570. John Wiley and Sons, 1992.

[BP92a] J. C. Bezdek and S. K. Pal. Cluster analysis. In J. C. Bezdek and S. K. Pal, editors, *Fuzzy models for pattern recognition - Methods that search for structures in data*, chapter 2, pages 29–35. IEEE Press, New York, NY, 1992.

[BP92b] Jim C. Bezdek and Nikhil R. Pal. *ClusPak v1.0 user's guide*. The IJK Company, University of West Florida, Pensacola, FL, July 1992.

[Car88] S. Carberry. Modeling the user's plans and goals. *Computational Linguistics*, 14(3):23–37, 1988.

[Car90] S. Carberry. *Plan recognition in natural language*. The MIT Press, Cambridge, MA, 1990.

[CH92] Jaime G. Carbonell and Philip J. Hayes. Natural language understanding. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 997–1015. John Wiley and Sons, 1992.

[CP79] P. R. Cohen and P. R. Perrault. Elements of a plan-based theory of speech acts. *Cognitive Science*, 3:177–212, 1979.

[ErH93] Henry C. Ellis and R. reed Hunt. *Fundamentals of Cognitive Psychology*. Brown and Benchmark, Madison, WI, 1993.

[Est78] William K. Estes. *Handbook of learning and cognitive processes*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1978.

[Fil68] Charles Fillmore. The case of case. In Emmon Bach and Robert T. Harms, editors, *Universals in Linguistic Theory*, pages 1–90, Chicago, IL, 1968. Holt, Rinehart and Winston.

[Fil71] Charles J. Fillmore. Types of lexical information. In D. D. Steinberg and L. A. Jakobovits, editors, *Semantics: an interdisciplinary reader*, pages 370–392. Cambridge University Press, London, UK, 1971.

[GD93] Avelino J. Gonzalez and Douglas D. Dankel. *The Engineering of Knowledge-Based Systems*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[Gre85] W. Green. *Computer-Aided Data Analysis: A Practical Guide*. Wiley-Inter science Publication, New York, NY, 1985.

[Gro81] B. J. Grosz. Focusing and description in natural language dialogues. In A. Joshi B. Webber and I. Sag, editors, *Elements of discourse understanding*, pages 85–105, Cambridge, England, 1981. Cambridge University Press.

[Har85] Mary Dee Harris. *Introduction to natural language processing*. Reston, VA, 1985.

[Kie92] D. E. Kieras. Cognitive modelling. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 176–181. John Wiley and Sons, 1992.

[McK83] K. R. McKeown. Focus constraints on language generation. In *Proceedings of the Third National Conference on Artificial Intelligence*, pages 582–587, Washington, DC, 1983.

[MS93] Yoshiro Miyata and Andreas Stolcke. Cluster 2.7. Design notes from electronic mail, 1993.

[NAS91] NASA. *CLIPS user's guide and reference manuals*. Software Technology Branch, L.B. Johnson Space Center, TX, sep 1991.

[NOA92] NOARL. *Database design document for the Naval Environmental Operational Nowcasting System (NEONS) version 3.5*. Naval Oceanographic and Atmospheric Research Laboratory, Monterey, CA, jun 1992.

[Pop86] Eduard V. Popov. *Talking with computers in natural language*. Springer-Verlag, New York, NY, 1986.

[PP94] Patrick Perrin and Frederick E. Petry. Knowledge discovery and information retrieval in large numerical databases: An application to ocean modeling and prediction. In M. M. Tanik W. Rossak and D.E. Cooke, editors, *Software Systems in Engineering PD-vol. 59*, pages 161–170, New York, NY, 1994. ASME.

[PSF91] Gregory Piatetsky-Shapiro and William J. Frawley. Knowledge discovery in databases: An overview. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in databases*, pages 1–27, Menlo Park, CA, 1991. AAAI Press / The MIT Press.

[Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[Rom84] C. Romesburg. *Cluster Analysis for Researchers*. Lifetime Learning Publications, Belmont, CA, 1984.

[Sea70] J. R. Searle. *Speech acts: an essay in the phylosophy of language*. Cambridge Univeristy Press, Cambridge, England, 1970.

[Sim73] Robert F. Simmons. Semantic networks: their computation and use for understanding english sentences. In R. Schank and K. M. colby, editors, *Computer models of thought and language*, pages 63–113. W. H. Freeman, San Francisco, CA, 1973.

[Tho77] P. W. Thorndyke. Cognitive strutures in comprehension and memory narrative discourse. *Cognitive Psychology*, 9:77–110, 1977.

[ZB91] Jan M. Zytkow and John Baker. Interactive mining of regularities in databases. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in databases*, pages 32–53, Menlo Park, CA, 1991. AAAI Press / The MIT Press.

DISTRIBUTION LIST

1. Scientific Officer (Code 1242)
   Office of Naval Research
   800 N. Quincy Street
   Arlington, VA 22217-5000
   - Mr. Robert Peloquin (3 copies)
   - Dr. Emanual Fiadeiro
   - Dr. Alan Weinstein

2. Administrative Grants Officer
   Office of Naval Research Resident Representative
   101 Marietta Tower-Suite 2805
   101 Marietta Street
   Atlanta, GA 30303

3. Director
   Naval Research Laboratory
   Attention: Code 2627
   Washington, DC 20375

4. Defense Technical Information Center
   Building 5, Cameron Station
   Alexandria, VA 22304-6145
   - Two Copies

5. Oceanographer of the Navy
   U.S. Naval Observatory
   34th and Massachusetts
   Washington, DC 20392

6. Naval Research Laboratory
   Code 7320
   Stennis Space Center, MS 39529

7. Technical Director
   Naval Oceanographic Office
   Stennis Space Center, MS 39529

8. Commanding Officer
   Fleet Numerical Oceanography Center
   Monterey, CA 93943-5000

9. Technical Director
   Naval Oceanography Command
   Building 1020
   Stennis Space Center, MS 39529

10. CDR David Markham
    Space and Naval Warfare Systems Command (PMW165)
    5 Crystal Park, Room 301
    Arlington, VA 22202

11. Manager
    University of Southern Mississippi
    Center for Ocean/Atmospheric Modeling
    Stennis Space Center, MS 39529

12. Engineering Research Center
    Mississippi State University
    Mississippi State, MS 39762
    - Dr. Joe Thompson
    - Dr. Robert Moorhead

13. Department of Computer Science
    Mississippi State University
    Mississippi State, MS 39762
    - Dr. Don Dearhold
    - Dr. Julia Hodges
    - Dr. Susan Bridges

14. Department of Computer Science
    Tulane University
    New Orleans, LA 70118
    - Dr. Fred Petry
    - Mr. Patrick Perrin

15. Center for Air Sea Technology
    Mississippi State University
    Stennis Space Center, MS 39529
    - Mr. Jim Corbin
    - Dr. Lanny Yeske
    - Mr. Don Goff
    - Mr. Ramesh Krishnamagaru
    - Mr. Valentine Anantharaj
    - Mr. Steve Foster

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. Agency Use Only *(Leave blank)*.	2. Report Date. 1 August 1994	3. Report Type and Dates Covered. TECHNICAL REPORT

4. Title and Subtitle. DESIGN OF AN INTELLIGENT SUPPORT SYSTEM FOR SCIENTIFIC DATABASES	5. Funding Numbers. *Program Element No.* *Project No.* *Task No.* *Accession No.*
6. Author(s). Patrick Perrin, Frederick Petry	
7. Performing Organization Name(s) and Address(es). Mississippi State University, Center for Air Sea Technology, Stennis Space Center, MS 39529-6000	8. Performing Organization Report Number. CAST TECHNICAL REPORT 94-2
9. Sponsoring/Monitoring Agency Name(s) and Address(es). Office of Naval Research, 800 North Quincy Street, Code 1242, Arlington, VA 22217-5000	10. Sponsoring/Monitoring Agency Report Number. REPORT 94-2

11. Supplementary Notes.
Research performed under Office of Naval Research Contract/Grant No. N00014-92-J-4109

12a. Distribution/Availability Statement. Approved for public release; distribution is unlimited	12b. Distribution Code.

13. Abstract *(Maximum 200 words)*.

This report describes progress made in designing an Intelligent Support System (ISS) for automatic data analysis and efficient data exploitation in numerical scientific databases, with an application to environmental databases used for ocean modeling and prediction. Improvements made to the Automatic Data Analysis System (ADAS) by combining the features of two machine learning techniques (i.e., data clustering and inductive learning by decision tree) to generate sets of production rules that efficiently describe the observational raw data contained in the scientific databases are explained. Data clustering allows the system to classify the raw data into clusters, which the system learns by induction to build the decision trees, from which are deduced the production rules. Also described is the design of a robust computational model of conversation (ENTRETIEN) for interactive natural language man-machine interfacing, based on case grammar theory. Such a system allows the user to naturally communicate with the system for efficient and effective information retrieval. ENTRETIEN conducts the conversation to recognize the user's intentions. To include the automatically generated production rules into the knowledge base of an expert system and to add the facilities of the natural language interface represent our final goal in building an intelligent support system for efficient exploitation of large scientific databases.

14. Subject Terms. (U) CAST (U) SCIENTIFIC (U) DATABASES (U) INTELLIGENT (U) ADAS (U) ENTRETIEN (U) ISS	15. Number of Pages. 34
	16. Price Code.

17. Security Classification of Report. UNCLASSIFIED	18. Security Classification of This Page. UNCLASSIFIED	19. Security Classification of Abstract. UNCLASSIFIED	20. Limitation of Abstract.